-

-

-

A [lock](http://msdn.microsoft.com/en-us/library/system.threading.monitor.aspx) **is specific to the AppDomain**,   
while [Mutex](http://msdn.microsoft.com/en-us/library/system.threading.mutex.aspx) **to the Operating System** allowing you to perform inter-process locking and synchronization (IPC).

[[](https://stackoverflow.com/users/29407/darin-dimitrov)](https://stackoverflow.com/users/29407/darin-dimitrov)

[Darin Dimitrov](https://stackoverflow.com/users/29407/darin-dimitrov)

**1.0m**267267 gold badges32713271 silver badges2921

-

-

-

60 vote:

A Mutex can be **either local** to a process **or system-wide**. [MSDN](http://msdn.microsoft.com/en-us/library/system.threading.mutex.aspx):

Mutexes are of two types: **local mutexes**, which are **unnamed**, and **named system mutexes**.

A **local** mutex exists only within your process.

Furthermore, one should take special care - detailed on the same page as well - when using a system-wide mutex on a system with Terminal Services.

One of the differences between Mutex and lock is that **Mutex utilizes a** [kernel-level construct](http://msdn.microsoft.com/en-us/library/ms684266(VS.85).aspx), so synchronization will always require at least a user space-kernel space transition.

**lock** - that is really a shortcut to the [Monitor class](http://msdn.microsoft.com/en-us/library/system.threading.monitor.aspx), on the other hand tries to avoid allocating kernel resources and transitioning to kernel code (and is thus leaner & faster - if one has to find a WinAPI construct that it resembles, it would be [CriticalSection](http://msdn.microsoft.com/en-us/library/ms682530(VS.85).aspx)).

The other difference is what others point out: a **named** Mutex **can** be used across processes.

Unless one has special needs or requires synchronization across processes, it is just better to stick to lock (aka Monitor)˛

There are several other "minor" differences, like how abandonment is handled, etc.

The same can be said about ReaderWriterLock and ReaderWriterLockSlim in 3.5, Semaphore and the new SemaphoreSlim in .NET 4.0 etc. It is true that the latter xxSlim classes cannot be used as a system-wide sync primitives, but they were never meant to - they were "only" meant to be faster and more resource friendly.

[Andras Vass](https://stackoverflow.com/users/276068/andras-vass)

**11.4k**11 gold badge3636 silver badges4949 bronze badges

[Add a comment](https://stackoverflow.com/questions/3735293/what-is-the-difference-between-lock-and-mutex)

-

-

-

113 vote:

lock is a compiler keyword, not an actual class or object. It's a wrapper around the functionality of the Monitor class and is designed to make the Monitor easier to work with for the common case.

The Monitor (and the lock keyword) are, as Darin said, restricted to the **AppDomain**.

Primarily because a reference to a memory address (in the form of an instantiated object) is required to manage the "lock" and maintain the identity of the Monitor.

The Mutex, on the other hand, is a .Net wrapper around an **operating system** construct, and can be used for system-wide synchronization, using string data (instead of a pointer to data) as its identifier. **Two mutexes that reference two strings in two completely different memory addresses,** **but having the same data, will actually utilize the same operating-system mutex**.

[[](https://stackoverflow.com/users/201070/toby)](https://stackoverflow.com/users/201070/toby)

[Toby](https://stackoverflow.com/users/201070/toby)

**7,294**33 gold badges2525 silver badges2626 bronze badges

[Add a comment](https://stackoverflow.com/questions/3735293/what-is-the-difference-between-lock-and-mutex)

-

-

-

32 vote:

I use a Mutex to check see if I already have a copy of the application running on the same machine.

bool firstInstance;

Mutex mutex = new Mutex(false, @"Local\DASHBOARD\_MAIN\_APPLICATION", out firstInstance);

if (!firstInstance)

{

//another copy of this application running

}

else

{

//run main application loop here.

}

// Refer to the mutex down here so garbage collection doesn't chuck it out.

GC.KeepAlive(mutex);

[[](https://stackoverflow.com/users/6910/jonathan)](https://stackoverflow.com/users/6910/jonathan)

[Jonathan](https://stackoverflow.com/users/6910/jonathan)

**25.5k**1313 gold badges6666 silver badges85

-

-

-